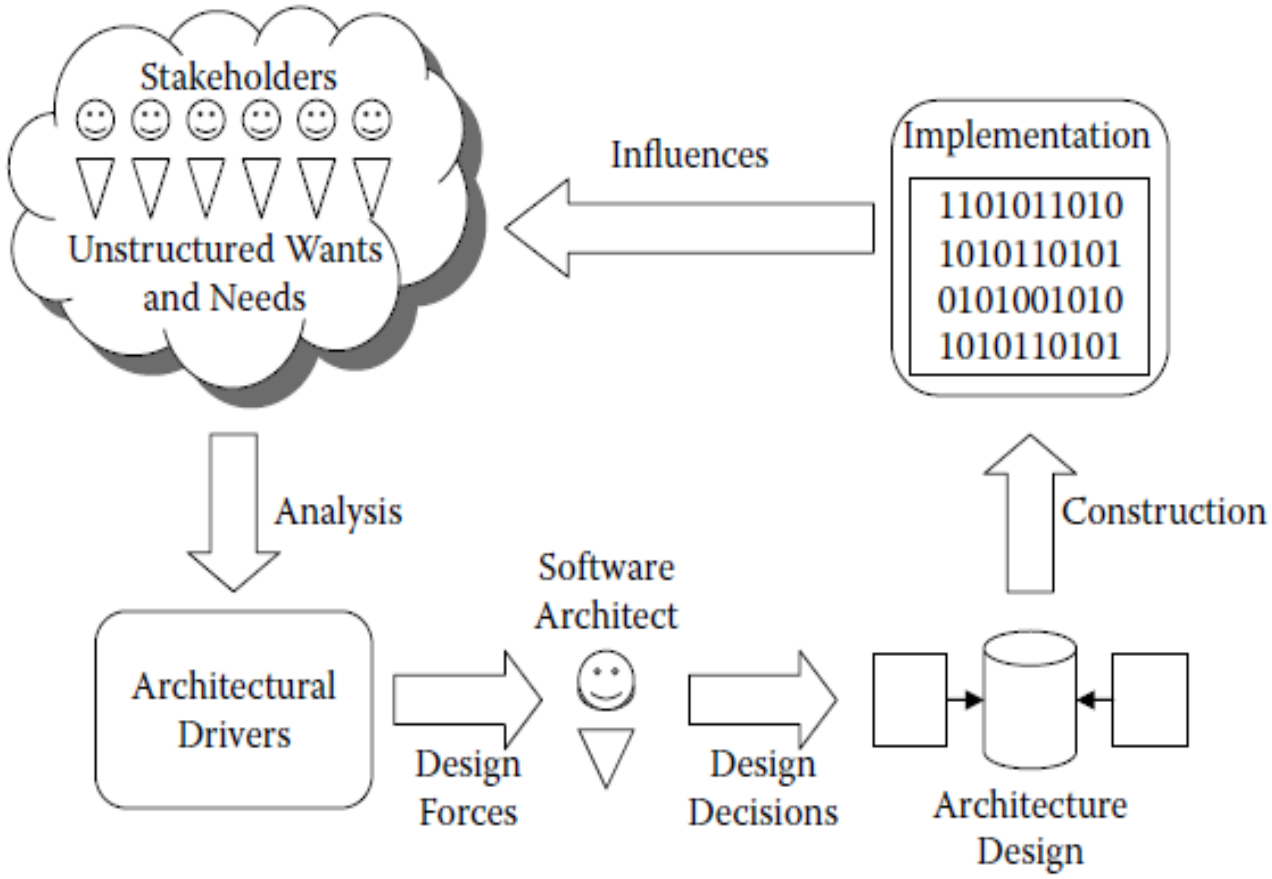
ПРОЦЕС ЗА ПРОЕКТИРАНЕ НА СОФТУЕРНАТА АРХИТЕКТУРА

Как започва проектирането на архитектурата?

- Проектирането започва при наличието на изисквания. От друга страна, не се изисква наличието на *много* изисквания, за да започне проектирането
- Архитектурата се оформя от няколко (десетина) основополагащи функционални, качествени и бизнес изисквания – т.н. архитектурни драйвери
- Ще разгледаме т.нар. процес Architecture Driven Design (ADD)

Цикличен процес на създаване на архитектурата



Как се избират драйверите?

1. Идентифицират се тези цели на системата, които са с най-висок приоритет
2. Тези цели се превръщат в сценарии за употреба или за постигане на качествено свойство
3. От тях се пробират не повече от 10 – тези, които имат най-голямо влияние върху архитектурата
4. След като драйверите бъдат избрани, започва проектирането. Започва и задаването на въпроси, което може да доведе до промяна в изискванията, което пък може да доведе до промяна в драйверите и т.н.

ADD – Attribute Driven Design

- ADD е подход за проектиране, в който основна роля играят качествените свойства (атрибути);
- Това е рекурсивен процес на дефиниране на архитектурата, като на всяка стъпка се използват тактики и архитектурни модели за постигане на желаните качествени свойства;
- В следствие на приложението на ADD се получават първите няколко нива на модулната декомпозиция и на други структури, в зависимост от случая
- Това е първото проявление на архитектурата и като такова е на достатъчно високо ниво, без излишни детайли

Стъпки на ADD

- Избира се модул, който ще се декомпозира. Изискванията към модула трябва да са известни.
- Модулът се детайлизира:
  - Избират се архитектурните драйвери (най-важните изисквания за този етап)
  - Избира се архитектурен модел, който удовлетворява драйверите. Модела се избира или създава въз основа на тактиките за постигане на избраните свойства. Идентифицират се типовете под-модули, необходими за постигането на тактиките
  - Създават се под-модули от идентифицираните типове и им се приписва функционалност съгласно сценариите за употреба. Създават се всички необходими структури.
  - Дефинират се интерфейсите към и от под-модулите
  - Проверяват се и се детайлизират изискванията, като същевременно се поставят ограничения върху под-модулите. На тази стъпка се проверява дали всичко съществено е налично и се подготвят под-модулите за по-нататъшна декомпозиция
- Това се повтарят за всички модули, които се нуждаят от по-нататъшна декомпозиция

Примерна система

- За илюстрация на ADD ще бъде разгледана примерна система:
- Продуктова линия за уреди за управление на гаражни врати (УУГВ), интегрирани с Домашна Информационна Система (ДИС)
- УУГВ е отговорен за отварянето и затварянето на гаражната врата и се задейства с бутон, дистанционно или от ДИС
- Освен това, ДИС включва функционалност за диагностициране на модула

Изисквания към примерната система (входни данни за ADD)

- Процесорите, контролерите и устройствата в различните продукти от продуктовата линия могат да се различават. Архитектурата за специфичен продукт трябва да може да се получава директно от архитектурата на продуктовата линия
- Ако вратата на гаража се удари в препятствие по време на затварянето, движението й трябва да спре в рамките на 100 ms
- Устройството трябва да бъде достъпно за управление и диагностика от ДИС, използвайки протокол, специфичен за продукта.

Входни данни на ADD

- Входните данни на ADD са набор от изисквания:
  - Функционални изисквания, под формата на сценарии за употреба (use-cases)
  - Функционални ограничения (constraints)
  - Качествени свойства, под формата на специфични сценарии за проявление

Стъпка #1: Избира се модул за декомпозиция

- Първоначално това е цялата система, която се разлага на подсистеми
- Подсистемите се разлагат на модули
- Модулите се разлагат на под-модули и т.н.
- В нашия случай – системата е УУГВ
- Примерно изискване – УУГВ трябва да се интегрира с ДИС

Стъпка #2.1: Избират се архитектурните драйвери

- Драйверите се избират измежду изискванията с най-висок приоритет
- В нашия случай, това са дадените по-рано 3 изисквания
- По някога въпросът дали дадено изискване е архитектурен драйвер или не не е тривиален и се налага детайлно изследване
- Напр., за да се види дали изискването за 100 ms време за реакция е важно от гледна точка на софтуерната архитектура се налага изследване на механиката на вратата и скоростта на процесорите
- Изборът на драйвери оформя декомпозицията на модула на под-модули. Въпреки, че има и други изисквания към модула, те се считат за по-маловажни и ще бъдат реализирани в контекста на най-важните

Стъпка #2.2: Избира се архитектурния модел

- За постигането на всяко изискване има тактика. Конфигурацията от избрани тактики определя архитектурния модел.
- Тактиките обикновено имат странични ефекти (+, -) върху другите свойства, затова се налага да се разгледа цялостната картина, т.е. архитектурния модел.
- В нашия случай драйверите са свързани с възможността за лесна промяна и с бързодействието

Стъпка #2.2: Продължение

- Тактиките за възм ожност за лесна промяна се делят на:
  - Локализиране на промените
  - Ограничаване на вълнообразните ефекти
  - Отлагане на обвързването
- В случая става въпрос за възможност за лесна промяна по време на проектирането, затова приложими са напр.:
  - Поддръжка на семантична съгласуваност
  - Скриване на информация

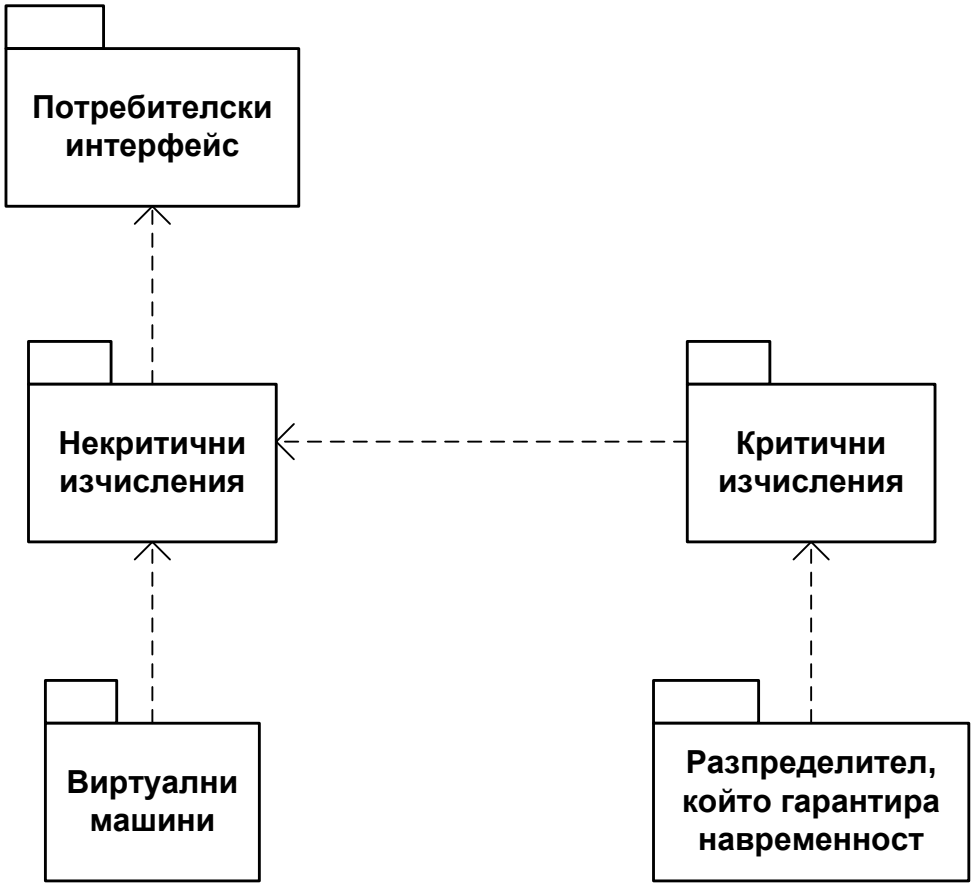
Стъпка #2.2: Продължение

- Тактиките за бързодействие се делят на:
  - Увеличаване на ефективността на алгоритмите
  - Намаляване на изискванията за ресурси;
  - Управление на ресурсите;
  - Управление на раздаването на ресурси;
- Приложими в случая са:
  - Увеличаване на ефективността;
  - Управление на раздаването на ресурси;

Стъпка #2.2: Продължение

- Общата картина на тактиките:
  - Поддръжка на семантична съгласуваност и скриване на информация – функционалностите, свързани с потребителския интерфейс, комуникациите и сензорите се отделят в собствени модули, т.н. “виртуални машини”. Очакванията са, те да се променят за всеки продукт от линията
  - Увеличаване на ефективността – изчисленията, свързани с критични към бързодействието активности трябва да са максимално ефективни
  - Разумно разпределение на ресурсите – ресурсите следва да се разпределят съгласно изискванията за навременност на критичните изчисления

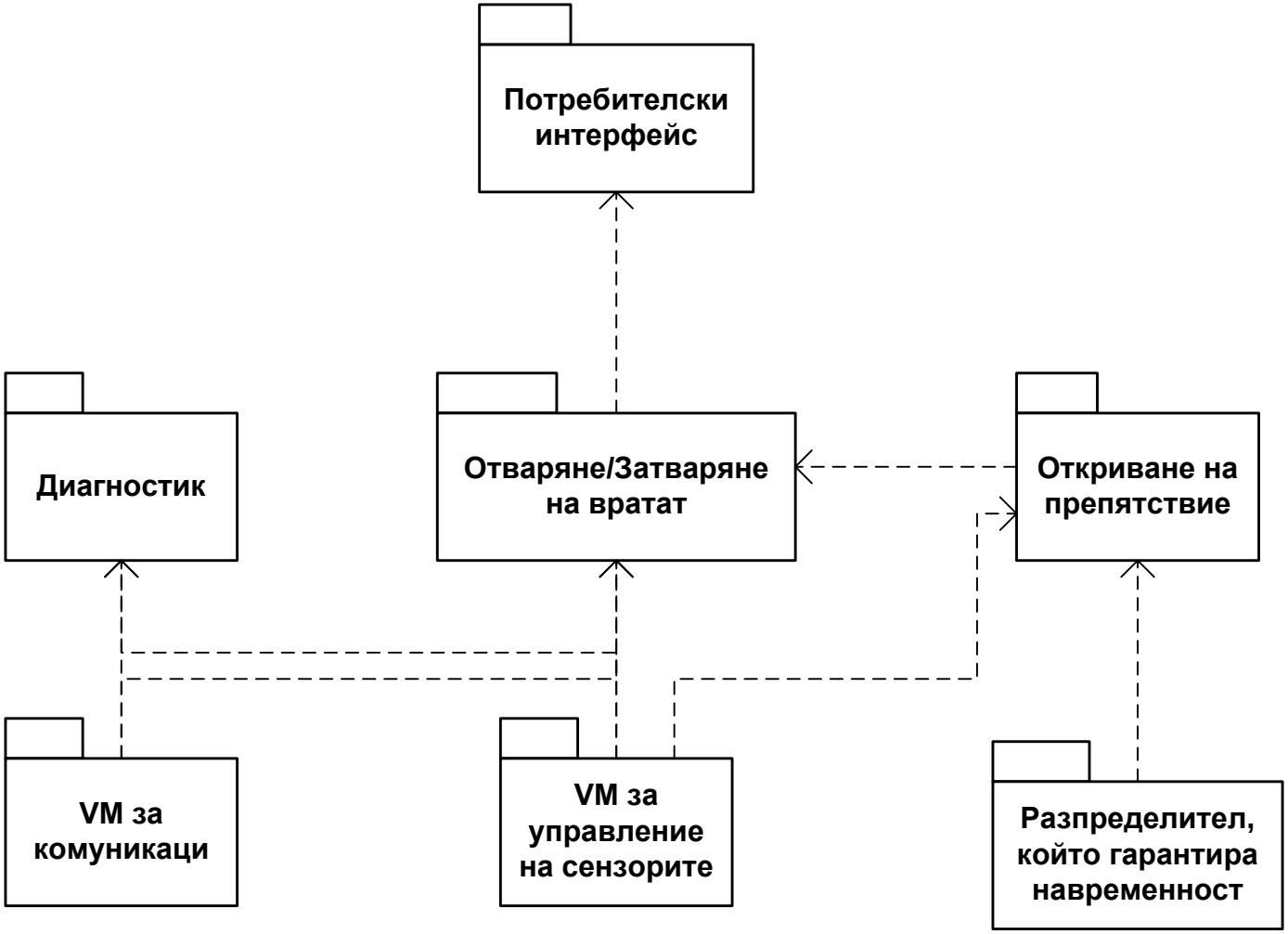
Стъпка #2.2: Архитектурен модел – идентифициране на модули



Стъпка #2.3: А. Създаване на под-модулите

- За всеки от идентифицираните типове се създават по един или няколко под-модула:
  - Откриването на препятствие и спиране на хода на вратата – критично изчисление
  - Отваряне/затваряне на вратата – некритично изчисление
  - Диагностиките – некритично изчисление
  - Две различни виртуални машини – едната за комуникациите, другата за работа със сензорите
  - Потребителския интерфейс, както и разпределителя са самостоятелни модули

Стъпка #2.3: Декомпозиция



Стъпка #2.3: Приписване на функционалност

- Всеки сценарий на употреба, който е приложим към модула, който декомпозираме се прилага, като при това се описва кои под-модули за коя част от функционалността отговарят
- В резултат се получават сценарии, приложими за под-модулите
- Възможно е в следствие на това да се премахнат или създадат нови под-модули
- По време на този процес изпъкват нуждите от обмяна на информация между под-модулите, които също трябва да бъдат документирани

Стъпка #2.3: Създаване на други структури

- За да стане ясно дали изискваната функционалност се покрива от под-модулите, се налага да бъдат разгледани и други структури, напр. на процесите и разположението
- Като за начало, обикновено се налага да се направи по една структура от всяка структурна група

Стъпка #2.3: Структура на декомпозицията

- Структура на декомпозицията – структурата, която директно се получава от метода. Получават се модули, на които се приписват функционалности и отговорности. Освен това структурата показва основните потоци на обмен на информацията

Стъпка #2.3: B. Структура на внедряването

- Наличието на няколко хардуерни устройства диктува необходимостта от структура на внедряването
- Логическите връзки се декомпозират и техните части се разполагат върху различните процесори, а между тях се оформят комуникационни канали
- Помага да се изясни дали са няколко инстанции на даден модул, дали е необходим специализиран хардуер
- Създаването на структурата се базира на архитектурните драйвери и избраните тактики за постигането на изискванията

Стъпка #2.4: Дефинират се интерфейсите на под-модулите

- Под интерфейс се разбира съвкупността от услуги и свойства, които модула предлага/изисква
- Документират се всички свойства и услуги от всички структури

Стъпка #2.5: Проверява се декомпозицията

- Чрез стъпките дотук се изгражда предложение за декомпозиция
- На последната стъпка се проверява дали тази декомпозиция е коректна, т.е. дали всички изисквания се покриват от нея
- Освен това, идентифицираните под-модули трябва да се приготвят за декомпозиция, ако това се налага

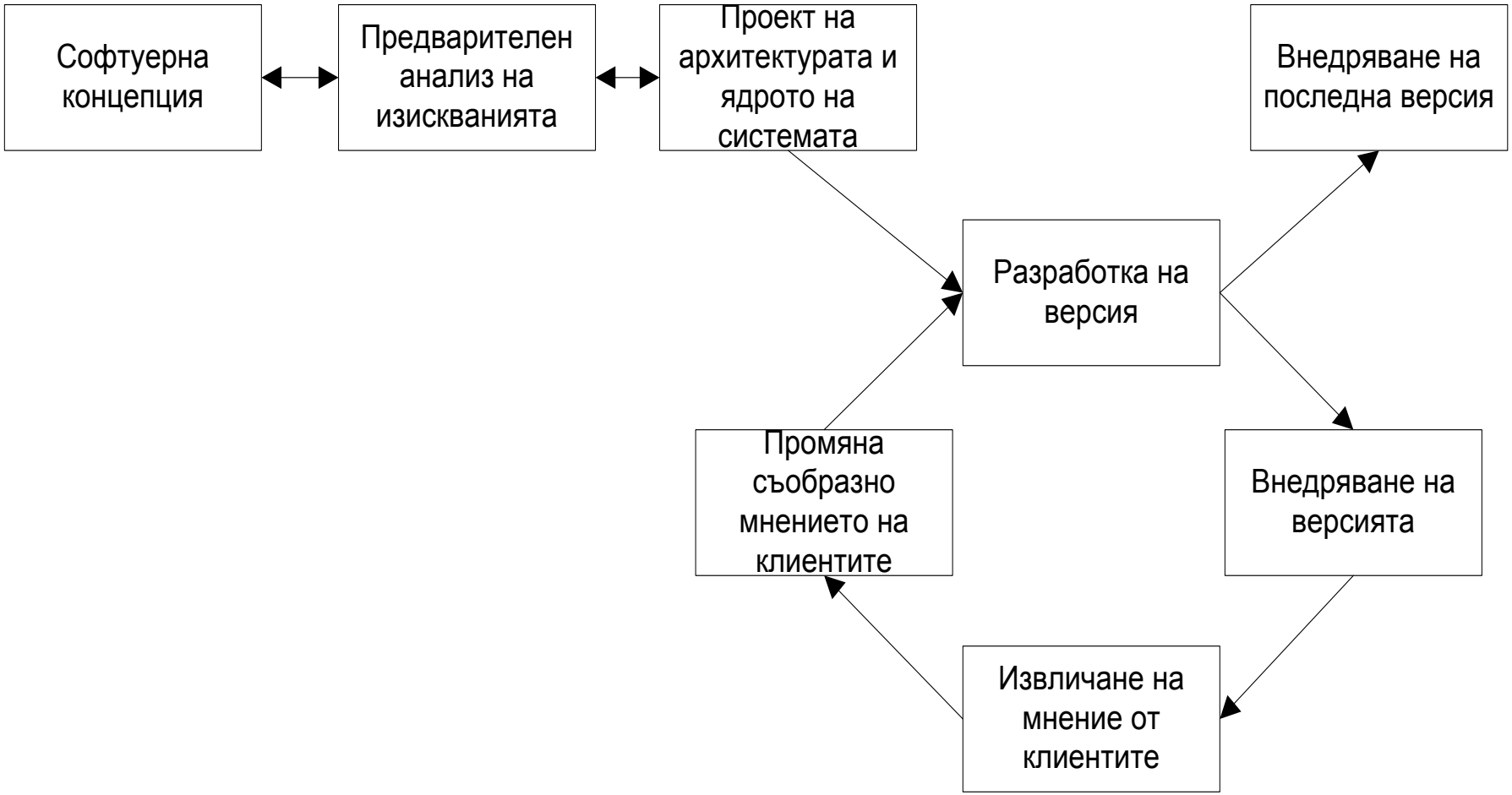
Стъпка #2.5: Проверява се декомпозицията 2

- Проверява се:
  - Как функционалните изисквания се изпълняват от под-модулите;
  - Дали се изпълняват ограниченията? За да се изпълнят ограниченията, може да се сложи отделен под-модул или да се вмени отговорността за това на един или повече под-модули;
  - Как се покриват сценариите за качествата? Напълно, с ограничения, без влияние или противоречат на декомпозицията?

Стъпка #3: Рекурсивен ADD

- След стъпка 2 имаме списък от под-модули, за които имаме:
  - Списък с отговорности;
  - Сценарии за употреба;
  - Интерфейси;
  - Сценарии за качества;
  - Списък с ограничения.
- Това е достатъчно за да послужи за вход на рекурсивно извикване на ADD за някои от модулите, които се нуждаят от детайлизация

Цикличен процес на създаване на архитектурата



3. Формиране на екипи

- След като се идентифицират първите няколко нива на декомпозицията, могат да се формират екипи, които да работят по съответните модули
- Структурата на екипите обикновено отговаря на структурата на декомпозицията
- Екипа представлява обособена екосистема, със собствени правила и експертиза
- Комуникацията в рамките на екипа е различна от комуникацията между екипите

Създаване на скелетна система

- Когато архитектурата е готова донякъде и има сформирани екипи може да се започне работа по системата
- По време на разработката обикновено се използват стъбове, за да могат модулите да се разработват и тестват поотделно
- Но с кои модули да започне създаването на скелетна система?

Последователност на реализацията

- Първо се реализират компонентите, свързани с изпълнението на и взаимодействието между архитектурните компоненти (middleware)
- След това се реализират някои прости функционалности
- Последователността по нататък може да се диктува от:
  - Намаляване на риска – първо най-проблематичните;
  - В зависимост от наличния персонал и квалификацията му;
  - Бързото създаване на нещо, което се продава;
- След това на база структурата на употребата се определят следващите функционалности